Автоматизация системного тестирования моделей аппаратуры на основе формальных спецификаций

М. М. Чупилко
chupilko@ispras.ru

Аннотация. В данной статье поднята проблема системного тестирования взаимосвязанных модулей аппаратуры, когда их сложность уже не позволяет применять подходы к тестированию на уровне модулей. В работе приводится краткий анализ возможностей построения тестовой системы на основе использования формальных спецификаций, а также предлагается метод верификации, являющийся расширением модульного подхода, основанного на технологии UniTESK.

1. Введение

Известно, что в проектах по разработке аппаратного обеспечения этап *верификации* является обязательным. Этот этап может быть достаточно длительным, около 70% от общего времени разработки [1]. Верификация позволяет проверить соответствие имеющейся аппаратуры (или ее модели) заявленной *спецификации*. В спецификации могут быть описаны разного рода требования: к функциональности, физическим характеристикам и др. Основным способом функциональной верификации является *тестирование* моделей аппаратуры.

В рамках данной статьи современный процесс производства компонентов аппаратуры можно условно разделить на три следующих этапа. Первый этап включает проектирование архитектуры будущего компонента, а также, возможно, создание упрощенной модели, описывающей процессы на системном уровне. Модель разрабатывается либо с использованием универсальных языков программирования (например, C/C++), либо с использованием специализированных языков системного уровня (SystemC [2], SystemVerilog [3] и др.).

Следующим этапом является создание модели на *уровне регистровых передач* (RTL[1]). На основе спецификации, иногда с привлечением общей модели системы, разрабатывается модель, содержащая детальное описание поведения компонента аппаратуры, которое может быть автоматически транслировано на *уровень вентилей*. На уровне вентилей в модели помимо функциональности начинает учитываться схемотехническая специфика интегральных схем. Имея модель на уровне вентилей, можно получать физические кристаллы, в этом заключается третий этап создания компонента, на котором функциональность создаваемого модуля не меняется, в данной статье мы его касаться не будем.

Теперь видно, что функциональные ошибки в аппаратуре могут быть выявлены уже в модели на уровне RTL, поскольку для получения физического компонента используются эквивалентные преобразования. Отсюда следует, что грамотно организованное тестирование моделей аппаратуры позволяет избежать функциональных ошибок в микросхемах. В случае сложных систем аппаратуры становится особенно актуальной тема автоматизации тестирования.

Компоненты аппаратуры различаются по своей сложности: они могут быть относительно простыми модулями, а могут представлять собой системы, в которых работает несколько функционально взаимосвязанных модулей. Для более качественного тестирования на каждом уровне «гранулированности» предлагается использовать различные подходы. Так, например, для тестирования модулей был предложен подход на основе формальных спецификаций [4]. Но для возможности его применения для более сложных компонентов требуется некоторая модификация метода.

Данная работа ориентирована на тестирование сложных моделей на уровне RTL. Статья содержит следующие разделы. В разделе «Существующие подходы к верификации систем» дается краткий обзор существующих технологий создания тестов для сложных моделей аппаратуры. Там же приводится обоснование использования формальных спецификаций. В разделе «Тестирование на модульном уровне» дается введение в существующий подход к функциональной верификации на модульном уровне, основанный на технологии UniTESK [5]. В разделе «Тестирование на системном уровне» предлагается подход, который продолжает традиции использования UniTESK, но уже применительно к многомодульным системам. В заключении приведены основные тезисы, характеризующие предлагаемый метод тестирования.

[1] RTL, Register Transfer Level, уровень регистровых передач. Для описания модели аппаратуры используются регистры, комбинационные блоки, их соединения между собой.

2. Существующие подходы к верификации систем

Подходы к верификации моделей аппаратуры можно разделить на два больших класса: формальную и имитационную верификацию. Формальные методы позволяют доказать математически, что модель работает корректно. Проверка осуществляется без учета динамики модели, с помощью анализа исходного кода. Существуют разные методы формальной верификации, однако они работают лишь для относительно простых моделей. Часто формальные методы используются для доказательства эквивалентности RTL-модели и модели на уровне вентилей. Для тестирования сложных моделей требуется слишком много ресурсов, и позволить себе такой подход могут только крупные фирмы. С увеличением вычислительной мощности компьютеров и совершенствования методик формальной верификации, возможно, они будут использоваться гораздо чаще, однако, на сегодняшний день их доля относительно мала.

Имитационная верификация подразумевает запуск модели с использованием симулятора. Для проведения верификации строится специальная тестовая система, которая генерирует стимулы для тестируемого компонента, проверяет реакции, полученные в ответ на поданные стимулы, а также оценивает полноту тестирования. Рассмотрим основные аспекты построения тестовых систем.

Тестовые последовательности могут быть описаны вручную, построены с использованием генератора случайных чисел, созданы на основе шаблонов, либо быть сгенерированы на основе обхода конечных автоматов. Ручное создание последовательностей не является технологичным, оно не применимо для тестирования сложных систем из-за больших требований к ресурсам и большой вероятности внесения ошибок в тесты: в системном тестировании необходима автоматизация создания стимулов. Одним из способов автоматизации является использование случайных стимулов: при таком подходе можно не тратить время на создание более «тонкой» методики и инструментов, но и вероятность нахождения сложных ошибок с его помощью довольно мала.

Другим часто используемым способом автоматизации является использование шаблонов. Шаблоны определяют некоторый класс эквивалентных ситуаций для проверки, создаются вручную и имеют вместо конкретных значений аргументов операций ограничения на их значения. Перед использованием шаблона в качестве тестового воздействия специальный компонент тестовой системы преобразовывает ограничения на значения параметров в систему уравнений и затем решает ее, используя техники разрешения ограничений. Если генерация случайных стимулов применяется, когда необходимо как можно раньше начать процесс тестирования, то использование техник разрешения ограничений позволяет нацелить тестирование на определенные классы функциональности. Подход к генерации стимулов на основе шаблонов используется в современных методологиях тестирования моделей

аппаратуры [6]. Необходимо учесть, что шаблоны все же создаются вручную со всеми вытекающими недостатками: ограничением на количество шаблонов, возможностью внесения в них ошибок, трудностью поддержки при изменениях в тестируемом компоненте. Поэтому провести полную верификацию компонента, а не только «крайних случаев», средствами шаблонов затруднительно.

Еще одним способом генерации тестовых воздействий является использование конечных автоматов. Конечный автомат можно представить как граф, дуги которого соответствуют подаче стимулов, а вершины представляют собой состояния тестируемой системы. Обычно для тестирования используются детерминированные автоматы, в которых для каждой последовательности входных воздействий (стимулов) существует лишь одно состояние, в которое автомат может перейти из текущего состояния. При обходе такого автомата происходит подача стимулов на тестируемую систему; процесс продолжается до покрытия всех вершин и дуг, либо до обнаружения ошибки.

Автоматы можно описывать явно с помощью графа состояний и переходов [7-9], можно описывать неявным образом, строя его на основе *формальной спецификации*, как это делается в технологии UniTESK. Явное описание автомата в реальных задачах сталкивается с трудностью поддержки его актуальности при постоянных изменениях в тестируемом проекте. В технологии UniTESK для описания каждой операции, которую реализует тестируемый компонент, задается *контракт*: *пред*- и *постусловия* операции. Контракт может накладываться не только на операцию в целом, но и на отдельные ее этапы. Неявное задание автомата осуществляется путем определения *функции вычисления текущего состояния* и описания *списка допустимых стимулов*. Построение автомата осуществляется динамически в процессе обхода. Для создания функции вычисления текущего состояния обычно используется информация о том, на каком этапе какая операция выполняется, а также информация о зависимостях между операциями. Таким образом, процедура поддержки автомата, заданного неявно с помощью формальных спецификаций, проще, чем описанного явно: при изменениях в проекте необходимо изменить спецификации, а тогда автоматически меняется и автомат.

Возвращаясь к проблеме построения тестовой системы, необходимо ответить на вопросы о *проверке правильности поведения* и *оценке полноты тестирования*. На оба вопроса позволяет ответить использование контрактных спецификаций: в первом случае это будет проверка всех необходимых постусловий на каждом этапе выполнения операций, во втором случае оценивается полнота обхода конечного автомата, заданного на основе формальных спецификаций – контрактов, и, фактически, описывающего устройство управления аппаратуры.

Альтернативными вариантами проверки правильности являются использование *эталонных моделей* и *самопроверяющихся тестов*.

Эталонные модели обычно используются для верификации в том случае, если они были разработаны на этапе архитектурного проектирования. Современным подходом к разработке эталонных моделей является использование транзакций [10] при передаче информации между модулями системы — такие модели называются *транзакционными*[2]. Отметим, что при использовании эталонных моделей есть как минимум два недостатка. Во-первых, разный уровень абстракции проверяемой и эталонной модели затрудняет проверку, а, во-вторых, для поддержки обоих моделей в актуальном состоянии при изменениях в проекте требуются значительные усилия. Конечно, формальные спецификации тоже нуждаются в поддержке, но их структура обычно проще: эталонные модели являются полноценными копиями проверяемого компонента, хоть и более абстрактными, а спецификации ограничиваются контрактами операций.

Самопроверяющиеся тесты подразумевают отсутствие какой-либо эталонной модели или спецификации, они содержат в себе проверки, которые необходимо осуществить после выполнения тестовой последовательности. Использование таких тестов вместе с автоматизированными генераторами тестовых последовательностей практически невозможно, поэтому они используются только при ручной разработке тестов, требуя значительных усилий по поддержке при изменениях в проекте.

Теперь осталось обсудить оценку полноты тестирования, которая может быть сделана двумя путями: использованием метрик на основе исходного кода модели, либо на основе спецификаций. Использование метрик покрытия кода очевидно и индикативно, но не отражает покрытие функциональности. Метрики на основе спецификации очень близки к покрытию функциональности при условии полного описания требований в спецификациях. Важным классом функциональных метрик являются так называемые *автоматные метрики*, определяемые на основе покрытия состояний и переходов конечных автоматов, построенных на основе требований. В контексте использования конечных автоматов в тестовых системах наиболее органичным видится и использование метрик на их основе для оценки полноты тестирования.

Описанные выше идеи использования формальных спецификаций и построения на их основе автоматов использовались при создании подхода к тестированию аппаратуры на модульном уровне [4], который успешно применяется для тестирования различных модулей промышленных микропроцессоров [11]. Поскольку у приведенных идей нет принципиальных

[2] Иногда говорят о модели на *уровне TLM* (*Transaction Level Modeling*, *моделирование на уровне транзакций*), когда имеется в виду более абстрактное представление аппаратуры по сравнению с уровнем RTL.

ограничений для использования на уровне систем (необходимо только избегать комбинаторного взрыва количества состояний конечного автомата), то их развитие для системного тестирования видится перспективным. К методу модульного тестирования и технологии UniTESK, на основе которой он разработан, обратимся в следующем разделе.

3. Тестирование на модульном уровне

Подход к модульному тестированию основан на технологии тестирования UniTESK. Как отмечалось в предыдущем разделе, при использовании данной технологии предполагается, что спецификация на тестируемый компонент записывается в виде контрактов: *пред-* и *постусловий* его операций. В случае тестирования программного обеспечения контракты накладываются на операции в целом, а в случае тестирования моделей аппаратуры каждая операция разбивается на микрооперации, для которых определяются индивидуальные контракты. На Рис. 1 изображено графическое представление спецификации на операцию: под *PRE* и *POST* понимаются пред- и постусловия микроопераций. Микрооперации могут быть записаны не только последовательно, но и с использованием ветвления потока управления операции [12].

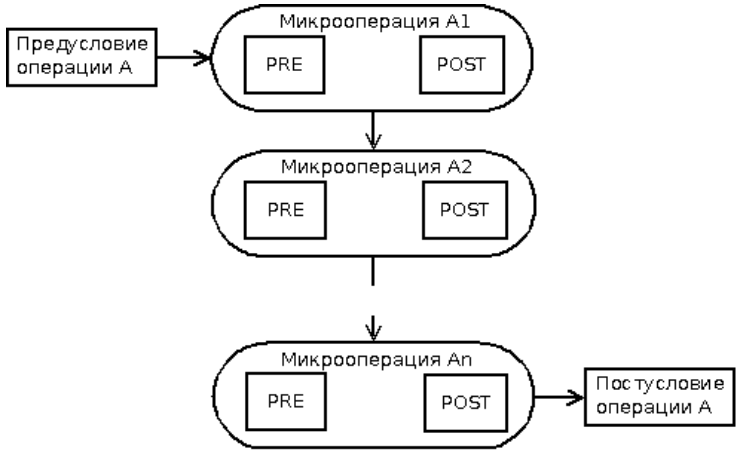

Рис. 1. Графическое изображение контракта на операцию.

Вычисление множества микроопераций, выполняемых параллельно на каждом такте (см. Рис. 2 и Рис. 3), используется как основа для задания текущего состояния автомата. Если предусловие операции выполнено, то она запускается, если выполнено предусловие микрооперации, то на следующем такте происходит проверка постусловия, а в состояние автомата добавляется информация о том, что данная микрооперация пройдена (например, *"A1*

finished"). Если микрооперация не выполнена, то в состоянии будет это отмечено (например, *"A3 delayed"*); если задержка происходит более одного такта, то состояние может отражать количество тактов задержки. После проверки постусловия осуществляется переход к следующей микрооперации рассматриваемой операции.

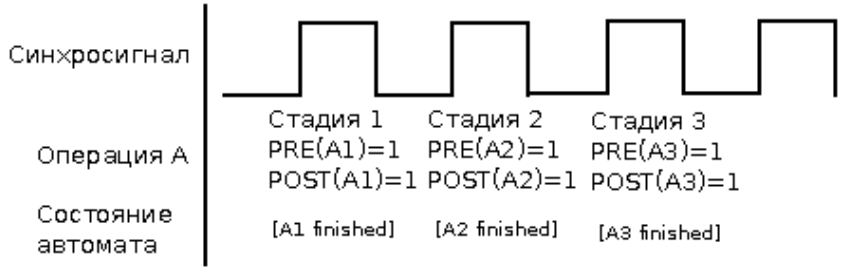

Рис. 2. Последовательное выполнение микроопераций одной операции.

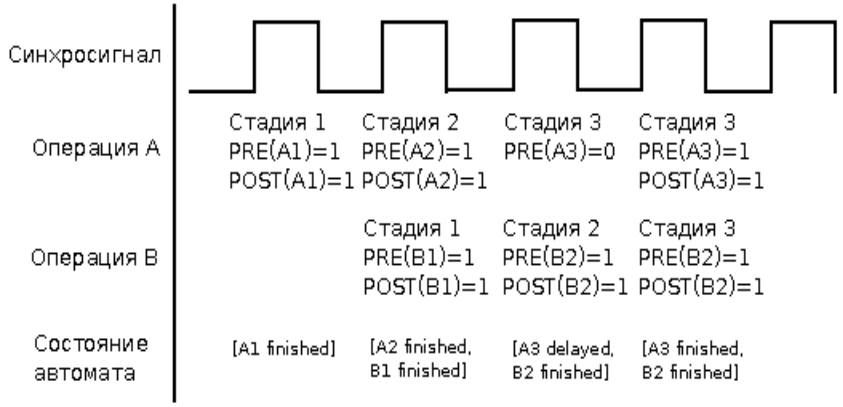

Рис. 3. Выполнение микроопераций на конвейере.

Отметим, что в случае асинхронных модулей принцип использования микроопераций остается, но меняется сигнал, к которому они привязаны: это уже не синхроимпульс *"CLK"*, а либо внешние сигналы, отвечающие за асинхронное взаимодействие, либо истечение заданного промежутка времени.

Данный метод уже несколько лет успешно применяется для тестирования моделей аппаратуры на модульном уровне. Он обеспечивает хорошее покрытие функциональности модулей с конвейерной архитектурой, находит в них ошибки после продолжительных испытаний самими разработчиками. Однако, у описанного подхода есть ограничения на сложность тестируемых модулей, поскольку их потактовая спецификация может привести к слишком большому числу состояний автомата. В следующем разделе будет предложена

модификация метода для возможности тестирования многомодульных систем, основанная на настройке точности тестирования.

4. Тестирование на системном уровне

Под *системой* мы будет понимать набор функционально зависимых модулей, имеющий единый внешний интерфейс, который служит для взаимодействия системы со своим окружением. Передаваемые через интерфейс операции для исполнения данной системой могут быть рассмотрены как на уровне всей системы, т.е. описаны посредством операций приема-передачи на уровне интерфейса системы, так и на уровне отдельных модулей, входящих в систему. Чем больше информации мы учитываем при спецификации той или иной операции, тем более тщательное тестирование становится возможным провести, тем большее количество различных ситуаций можно создать. Будем называть описание выполняемых операций, при котором не учитываются внутрисистемные особенности выполнения операции, *описанием на уровне системы*. Описание операций, в котором отведено место более детальному анализу межмодульного поведения операции, будем называть *описанием на уровне модулей*. В некоторых случаях необходимо учитывать также особенности поведения операции внутри модулей, в этом случае будем говорить о *субмодульном уровне*.

В процессе выполнения любой операции, на каком бы уровне она ни была рассмотрена, имеются этапы обмена данными между некоторыми сущностями (блоками, модулями, подсистемами), имеющими смысл на данном уровне. При рассмотрении уровня системы обмен информацией может быть только с окружением системы. При переходе на модульный уровень появляется информация об этапах обмена данными между модулями. Каждый этап такого взаимодействия легко фиксируется и лежит в основе спецификации видимого поведения операции. Будем называть этапы выполнения операции, в которых происходит какое-либо интерфейсное взаимодействие, *точками синхронизации*.

Иногда может возникнуть необходимость рассмотреть какие-либо дополнительные этапы выполнения операции, наличие которых на данном уровне абстракции (например, при рассмотрении только системного уровня) не следует из имеющегося описания интерфейсных взаимодействий. Даже на модульном уровне может возникнуть необходимость учесть некоторые особенности поведения тестируемого компонента, которые реализованы на уровне блоков. Тогда верификатору совместно с инженером-разработчиком необходимо расширить список точек синхронизации *дополнительными точками*. Описать такие точки можно либо с помощью количества тактов, которые должны пройти с момента последней точки синхронизации данной операции, либо с помощью использования интерфейсных сигналов, скрытых на данном уровне рассмотрения системы (поскольку сейчас идет речь о тестировании моделей, данный прием всегда возможен).

Основная причина, ограничивающая применение подхода к модульному тестированию для верификации многомодульных систем, заключается в комбинаторном взрыве количества состояний в конечном автомате вследствие использования слишком детальных спецификаций. Для перехода к системному тестированию необходимо уменьшить детализацию. В качестве средства настройки точности тестирования предлагается использовать точки синхронизации. Далее детально рассматриваются вопросы задания конечного автомата на основе точек синхронизации.

4.1. Построение конечного автомата для систем аппаратуры

В процессе модульного тестирования состояние конечного автомата, описывающего тестируемый компонент, обновлялось на каждом такте (см. Рис. 2 и Рис. 3). При этом учитывалась информация о выполненных микрооперациях, задержанных микрооперациях, а также о времени их задержки, и, кроме того, могла учитываться дополнительная информация, например, о зависимостях между операциями. В результате число возможных состояний автомата при тестировании сложных систем получается слишком большим.

Для уменьшения числа состояний предлагается перейти от потактового вычисления текущего состояния на основе микроопераций, к построению состояния на основе точек синхронизации. В таком случае контракты (предусловие соответствует предикату достижимости точки, постусловие – проверке результатов интерфейсного взаимодействия) накладываются не на микрооперации, а на точки синхронизации. Соответственно, формальная спецификация теперь включает в себя только ключевые аспекты важные для межмодульного взаимодействия. Автомат, который учитывает только точки синхронизации, содержит существенно меньше состояний, поскольку он не берет в расчет задержки выполнения этапов операций и сведения о субмодульном выполнении операций.

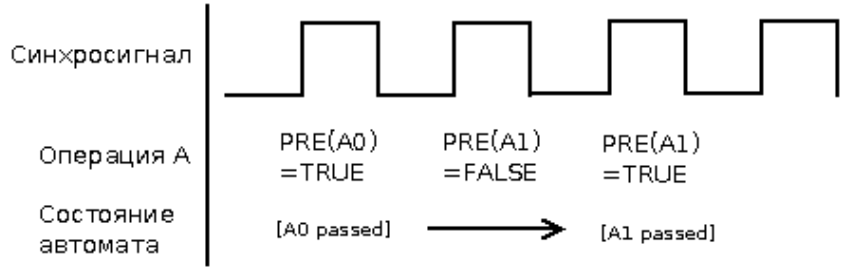

Рис. 4. Пример перехода в автомате при выполнении одной операции.

На Рис. 4 проиллюстрировано, как определяется момент времени, соответствующий переходу в очередное состояние автомата. Для наглядности рассмотрен случай, когда выполняется только одна операция. В данном

примере точка синхронизации *A0* достигается сразу, точка синхронизации *A1* задерживается на 1 такт, в котором функция вычисления текущего состояния модуля не вызывается. Следовательно, в тестовую систему вводится дополнительный компонент, который работает на каждом такте и проверяет, достигнута или нет очередная точка синхронизации. Достижение точки соответствует «окончанию» перехода в следующее состояние конечного автомата, описывающего данную операцию. По достижении точки синхронизации можно подавать новые стимулы для продолжения обхода автомата.

Если выполняется одновременно несколько операций, что обычно и наблюдается в системах аппаратуры, то появляется понятие *глобального автомата* (в дальнейшем слово *глобальный* будет опускаться, если это не будет вносить путаницы), который в общем случае содержит композицию автоматов выполняющихся операций. При построении композиции автоматов достигнутые на данном такте точки синхронизации будут помечаться как *passed*. Например, текущее состояние глобального автомата *[A0, B0 passed]* говорит о том, что сейчас работают 2 операции *A* и *B*, на данном такте достигнута точка синхронизации *B0*, а ближайшая точка синхронизации операции *A* – это *A0*.

В случае если тестируемая система допускает параллельную обработку нескольких операций, построить детерминированный автомат только на основе только точек синхронизации без учета дополнительной информации о зависимостях между операциями не всегда возможно. На Рис. 5 приведен пример, в котором операция *A* в точке синхронизации *A0* и *D* в точке *D0* влияют на время появления точки *B2*, но при этом *C1* сохраняет свое положение. В результате из состояния *[B1 passed, C1]* мы попадаем либо в *[B2 passed, C1 passed]*, либо в *[B2, C1 passed]* недетерминированным образом. В данном случае помогло бы введение дополнительной точки *D_1*, которая бы символизировала освобождение некоторого разделяемого ресурса, который не поделили операции *B* и *D*.

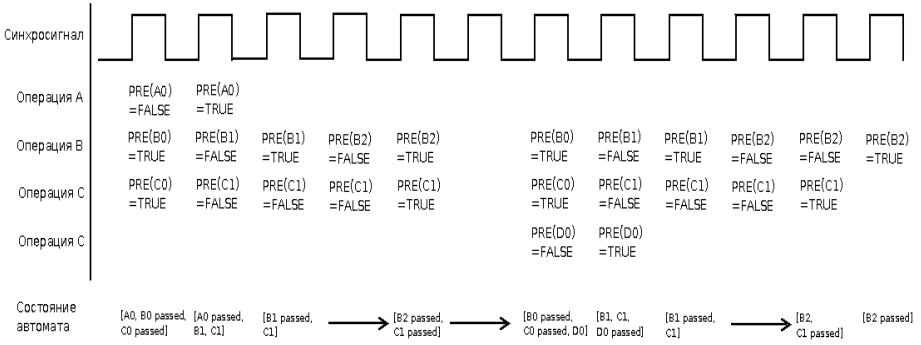

Рис. 5. Недетерминизм автомата, основанного на точках синхронизации.

Следовательно, описание глобального детерминированного автомата в случае параллельных операций должно строиться на основе точек синхронизации и, при необходимости, дополнительных точек. Оставим гипотезу о том, что в общем случае точек синхронизации и дополнительных точек достаточно для описания детерминированного автомата, без доказательства.

В примере, приведенном на рис. 5, предполагается, что каждая новая операция может быть запущена в точках синхронизации предыдущих операций или при пустом состоянии конвейера. Такой подход возможен, однако, во-первых, необходимо достаточно тонко обходить возможный недетерминизм автомата, а, во-вторых, в результате тестирования получается одна большая трасса теста, которая может быть сложна для нахождения короткого пути для повторения однажды возникшей ошибки. Поэтому в ряде случаев предлагается использовать немного модифицированные правила запуска стимулов и учета точек синхронизации работающих операций.

Так, предлагается «фиксировать» одну из выполняемых тестируемым компонентом операций, исполнение которой разрешено начинать только при пустом состоянии конвейера. Все остальные операции можно запускать только в процессе выполнения фиксированной, но как только фиксированная операция завершается, допускается только завершение всех параллельных операций и приход к пустому состоянию конвейера (Рис. 6). Также предлагается не учитывать в состоянии автомата точки синхронизации всех параллельных (по отношению к фиксированной) операций; тогда производится учет только списка операций, которые были поданы в точках синхронизации фиксированной.

Рис. 6. Использование коротких тестов.

Таким образом, тестирование разбивается на набор коротких тестов, с помощью которых в точках синхронизации фиксированной операции тестируются старты всех совокупностей допустимых стимулов. Основным преимуществом этого способа является достаточно легкое воспроизведение найденной ошибки.

В случае учета точек синхронизации операций, параллельных фиксируемой, появляются те же проблемы, связанные с недетерминизмом автомата, как и в случае «длинного» теста. Но при этом же получается набор коротких тестов, полная совокупность которых будет эквивалентна тестовой последовательности, полученной путем обхода автомата в «длинном» тесте.

Подводя итоги данного подраздела, можно сказать, что детерминированный конечный автомат для тестирования систем должен строиться не только на основе точек синхронизации, но и путем учета дополнительных точек. В состоянии автомата учитываются как достигнутые на данном такте точки, так и те, которые еще предстоит достичь во всех остальных работающих на данном такте операциях. Построенный таким образом автомат может быть единым для всего теста, а может разбиваться на набор маленьких автоматов для получения коротких тестов. В следующем подразделе попробуем расширить возможности подачи стимулов в промежутках между точками синхронизации.

4.2. Расширение возможностей подачи стимулов

В приведенном выше способе создания конечного автомата для системного тестирования предполагалось, что подача стимулов осуществляется только при достижении некоторой точки синхронизации (или дополнительной точки). В промежутках же между точками стимулы не подаются.

В некотором смысле моменты времени между двумя соседними состояниями глобального автомата можно считать *эквивалентными* и осуществлять подачу тестовых воздействий на них, а не только при достижении точек синхронизации. Такт для подачи в этом случае выбирается случайным образом. Однако при подаче стимулов в таких промежутках можно нарушить требование детерминизма автомата. Приведем пример. Пусть при выполнении операции *A* возникло два последовательных промежутка: *A0* (*x1* тактов) и *A1*. Промежуток *B0* (*x1/2* тактов) операции *B* стартует в одном случае в начале *A0*, а в другом – в конце. Тогда при одинаковом исходном состоянии *[A0 passed]* следующее состоянии автомата может быть либо *[A1 passed, B0]*, либо *[A1 passed, B0 passed]*. Очевидно, что в данном случае необходимо использовать дополнительную информацию: например, на каком конкретно такте из промежутка была запущена новая операция. Но учет такой информации может привести к росту количества состояний. Также необходимо учитывать, что точки синхронизации выбираются не случайным образом, поэтому их покрытие должно осуществляться в первую очередь. Следовательно, возможностями подачи стимула в эквивалентных промежутках следует пользоваться с осторожностью.

При повышении уровня абстракции неизбежно должны потеряться некоторые случаи, которые можно было бы охватить на модульном уровне. Рассмотрим такие случаи в следующем подразделе.

4.3. Сравнение модульного и системного подходов

Тестирование на модульном уровне в основном отличается от тестирования на системном уровне детализацией проверок и детализацией состояний. В случае модульного подхода, например, учитывается такая характеристика, как количество тактов, прошедших после блокировки микрооперации, т.е. разным

задержкам в автомате будут соответствовать разные состояния, соответственно в каждом из них (на каждом такте задержки) будет осуществляться попытка подать очередной стимул. Понятно, что некоторые из тактов ожидания можно считать эквивалентными, но если разработчик тестов не знает особенностей реализации блокировок в тестируемом модуле, такой избыточный подход оказывается предпочтительным, поскольку позволяет покрыть все ситуации.

В случае системного подхода такты ожидания не учитываются, а стимул подается либо в точке синхронизации, либо случайным образом во временном промежутке между такими точками. Поэтому, если между точками синхронизации происходила какая-либо важная скрытая активность, она не будет гарантированно проверена в сочетаниях с другими операциями (см. рис. 7). Таким образом, для повышения качества тестирования верификатору необходимо выяснять все интересные точки, в которых происходят какие-либо важные скрытые процессы в модулях в составе системы. Этими интересными точками необходимо расширять списки точек синхронизации и дополнительных точек в используемых тестах и получать возможность изменения детализации тестирования, сохраняя баланс между качеством и временем тестирования.

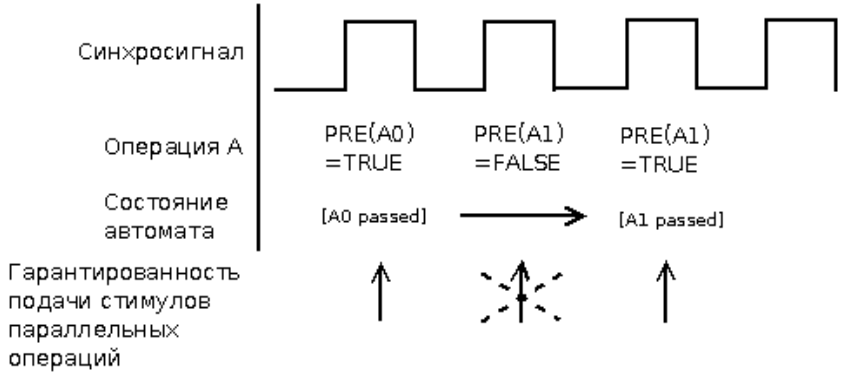

Рис. 7. Меньшая детализация тестирования в случае системного подхода.

5. Заключение

В данной работе были представлены основы метода автоматизированного системного тестирования моделей аппаратуры, использующего формальные спецификации так называемых *точек синхронизации* операций. Апробация данного метода пока не осуществлялась, но, учитывая, что в его основе лежит метод тестирования на модульном уровне, переработанный в соответствии с реалиями систем, автору видится перспективность его использования.

Инструментальная поддержка подхода может быть осуществлена с использованием инструментов технологии UniTESK.

Литература

[1] J. Bergeron. Writing Testbenches Using SystemVerilog. Springer Media, 2006.
[2] IEEE Standard SystemC Language Reference Manual. IEEE Std 1666-2005
[3] IEEE Standard for SystemVerilog — Unified Hardware Design, Specification, and Verification Language. IEEE Std 1800-2009.
[4] А.С. Камкин, Метод автоматизации имитационного тестирования микропроцессоров с конвейерной архитектурой на основе формальных спецификаций, диссертация на соискание уч.ст. к.ф.-м.н., Москва, 2009.
[5] http://www.unitesk.ru
[6] S. Iman. Step-by-step Functional Verification with SystemVerilog and OVM. Hansen Brown Publishing Company, 2008.
[7] S. Ur, Y. Yadin. Micro architecture coverage direction generation of test programs. Proceedings of DAC, 1999.
[8] P. Mishra, N. Dutt. Functional coverage driven test generation for validation of pipelined processors. Proceedings of DATE, 2005.
[9] R. Ho, C. Yang, M. Horowitz, D. Dill. Architecture validation for processors. Proceedings of ISCA, 1995.
[10] TLM-2.0 Reference Manual, http://www.systemc.org/downloads/standards/
[11] M. Chupilko, A. Kamkin, D. Vorobyev. Methodology and Experience of Simulation-Based Verification of Microprocessor Units Based on Cycle-Accurate Contract Specifications. Proceedings of SYRCOSE, 2008.
[12] M. Chupilko, A. Kamkin. Specification-Driven Testbench Development for Synchronous Parallel-Pipeline Designs. Proceedings of NorChip, 2009.